Chancen und Risiken LDAP-basierter zentraler Authentifizierungssysteme

Peter Gietz
DAASI International GmbH
Wilhelmstr. 106
D-72074 Tübingen
peter.gietz@daasi.de

Zusammenfassung

LDAP wird zunehmend für Authentifizierungsprozesse verwendet und hat sich diesbezüglich zu einem Standard-Mechanismus entwickelt. Viele Anwendungen besitzen eine integrierte LDAP-Schnittstelle für die Authentifizierung von Benutzern und auch Login-Prozesse können über LDAP abgewickelt werden. Dies hat den Vorteil, dass ein Benutzer sich nur ein einziges Passwort merken und der Administrator nur eine einzige Benutzerverwaltung administrieren muss. Allerdings wird die Kompromittierung des zentralen Passworts auch zu einem erhöhten Sicherheitsrisiko, weshalb die Sicherung der in LDAP gespeicherten Passwörter eine wesentliche Voraussetzung für LDAP-basierte zentrale Authentifizierungssysteme ist. Der Beitrag führt in das Thema ein, beleuchtet die diesbezüglichen Bedrohungsszenarien, sowie die verschiedenen möglichen Abwehrmaßnahmen.

1 Einführung Identity Management und Provisioning

Moderne IT-Infrastrukturen stellen Benutzern Ressourcen (z.B. Zugriff auf Rechner, Drucker etc.) und Dienste (z.B. Informationsdienste) über Computernetzwerke zur Verfügung. Um zu steuern, welche Benutzer auf welche Ressourcen und Dienste zugreifen dürfen, werden Authentifizierungs- und Autorisierungsprozesse implementiert, wobei die Authentifizierung den Beweis einer Identität darstellt und die Autorisierung einer Identität Zugriffsrechte, die meist in sogenannten *Access Control Lists* (ACLs) spezifiziert werden, zuweist und durchsetzt. In diesem Zusammenhang kann Identität als eine eindeutige Kennung definiert werden, die eine Person gegenüber einem Computersystem identifiziert, z.B. eine Login-Id die einen Zusammenhang mit einer Person darstellt. Eine Person kann in verschiedenen Zusammenhängen verschiedene Identitäten haben, entweder auf unterschiedlichen Computersystemen, oder in unterschiedlichen Rollen bei einem Computersystem. Auch andere Entitäten als Personen können in diesem Sinne Identitäten sein, z.B. Computerprogramme, Computer, etc. Prozesse zum Verwalten von Identitäten werden *Identity Management*[1] genannt. Mit Identity Management wird erreicht, dass Personen Informationen über sich veröffentlichen, um z.B. kontaktiert werden zu können;

1 Gute Einführungen in Identity Management und Provisioning findet man u.a. in [Pohlman 2003] und [OpenGroup 2002].

Informationen über andere Personen erhalten; sich authentifizieren, also ihre Identität beweisen, um Ressourcen und Dienste in Anspruch nehmen zu können;im Netz bezahlen können. Organisationen können durch Identity Management Identitätsinformationen über Mitarbeiter oder Mitglieder verwalten; Benutzer ihrer Ressourcen verwalten; Konsistenz der Identitäten in verschiedenen Informationsspeicher erreichen; Vortäuschung falscher Identitäten verhindern.

Folgende organisatorische Prozesse werden nach [OpenGroup 2002] im Identity Management abgebildet:

- Personen werden in Organisationen aufgenommen, erhalten Rollen und Berechtigungen, agieren in ihrer Rolle, wechseln Rollen und Berechtigungen, verlassen die Organisation
- Organisationen bzw. Organisationseinheiten werden gegründet, agieren in Arbeitsprozessen, werden zusammengefügt (*merge),* werden aufgeteilt (*split*), werden wieder aufgelöst

Solche organisatorischen Prozesse werden im Identity Management mit folgenden IT-Prozessen abgebildet:

- Identitäten erzeugen
- Identitätsinformationen aktualisieren
- Identitäten löschen
- Identitäten archivieren
- Identitätsinformation anfordern und anzeigen
- Identitäten verifizieren
- Mit Identitäten signieren (PKI)
- Zugriffskontrollregeln durchsetzen (Lese- und Schreibrechte)
- Datenbanken für Identitäten aufbauen und pflegen
- Identitätsdatenbanken synchronisieren
- Identitätsdatenbanken aufteilen und zusammenführen

In veralteten IT-Landschaften finden solche Prozesse in verschiedenen Systemen und Anwendungen jeweils mit eigenen Technologien statt. Gerade in größeren Infrastrukturen, in denen viele Ressourcen und Dienste zur Verfügung gestellt werden, macht es Sinn, Identity Management Prozesse zentral mit einer einzigen Technologie abzuwickeln und die entsprechenden Informationen bzw. Prozesse über sogenanntes *Provisioning* an die einzelnen Ressourcen und Dienstanwendungen weiterzugeben. Provisioning wird nach [Pohlman 2003, S. 56] wie folgt definiert:

> *"Provisioning is an emerging directory-guided IT solution that automates the deployment of IT resources and services based on business requirements"*

Hierbei wird die zentrale Rolle von Verzeichnisdiensten bei diesen Prozessen hervorgehoben ("*directory guided"*). Verzeichnisdienste sind in diesem Zusammenhang so wichtig, weil sie eine standardisierte Netzwerkschnittstelle zur Verfügung stellen, über die verschiedene Anwendungen gleiche Dienste wie Identitätsfeststellung, Rollenzuweisungen, etc. in Anspruch nehmen können. Es handelt sich hierbei also um eine Softwareschicht, die über standardisierte Schnittstellen im Netz die gleichen Dienste für unterschiedliche Anwendungen zur Verfügung stellt. Diese Softwareschicht wird auch *Middleware* genannt[2]. Unter diesem Begriff fallen sehr unterschiedliche Dienste, wie z.B. CORBA (Common Object Request Broker Architecture)[3], eine Objektschnittstelle für Anwendungen, sowie *Web Caching*[4], aber

2 Zum Begriff Middleware vgl. [RFC 2768], sowie

3 Aktuellste Spezifikation ist [OMG 2002].

4 Vgl. z.B.: [Wessels 2001]

eben auch Verzeichnisdienste, PKI und Authentifizierungssysteme. Das neue Paradigma des Internets *Web Services*[5], welches gegenwärtig vom W3C-Konsortium[6] spezifiziert wird, ist ohne das Konzept Middleware undenkbar.
Im Folgenden wird untersucht, wie Verzeichnisdienste, insbesondere solche, die auf dem internationalen IETF-Standard LDAP beruhen, im Rahmen von Identity Management eingesetzt werden können.

2 Kurzdarstellung von LDAP

LDAP (*Lightweight Directory Access Protocol*)[7] hat sich als ein internationaler Standard für Verzeichnisdienste durchgesetzt und wird zur Bereitstellung von verschiedensten Informationsdiensten im Netz verwendet. Es gibt mittlerweile internationale standardisierte Daten-Schemata insbesondere zur Abbildung von Personen, also z.B. Kontaktdaten wie Email-Adresse und Telefonnummern[8] aber auch Benutzer-Account-Daten wie Login-ID und Passwort[9], sowie Zertifikate, die im Rahmen von PKIs (*Public Key Infrastructure*) Verwendung finden[10]. Aber auch zur Abbildung von Organisationsstrukturen, sowie verfügbarer Ressourcen, wie Rechenleistung, Speicher und Drucker gibt es einheitliche Schemata[11].
LDAP besteht im Wesentlichen aus einem Netzwerkprotokoll (vergleichbar mit HTTP oder FTP) sowie aus einem objektorientierten Datenmodell, welches aus dem X.500-Standard [X.500] übernommen worden ist. Beide können hier nur kurz angerissen werden.[12]
Das Netzwerkprotokoll verfügt über alle Operationen, die für das Datenmanagement (*add, delete, modify, modifyDN*) und für Datenretrieval (*search*) notwendig sind. Letztere Operation ist sehr mächtig, da neben einem beliebig komplexen Suchfilter u.a. auch die zurückzugebenden Attribute, der Ort im Datenbaum, ab dem die Suche anfangen soll (der sog. *Base DN*), sowie die Anzahl der Hierarchieebenen, die durchsucht werden sollen (nur der durch den BaseDN spezifizierte Eintrag, alle genau eine Ebene unter diesem Eintrag liegenden Einträge, oder der gesamte durch den BaseDN spezifizierte Teilbaum) spezifiziert werden können. Des weiteren werden Operationen für Authentifizierung (bind) und das Abmelden vom Server (*unbind, abandon*) spezifiziert. Der Authentifizierungsvorgang in LDAP wird genauer im nachfolgenden Kapitel behandelt. Das Netzwerkprotokoll kann durch im Standard spezifizierte Erweiterungsmechanismen mit neuen Funktionalitäten versehen werden, wobei entweder vorhandene Operationen erweitert, oder ganz neue Operationen definiert werden.
Das Datenmodell baut auf sog. Attributen auf, in der die gesamte Information eines Verzeichniseintrag genannten Informationsobjekts abgelegt werden kann. Ein Attribut besteht aus einem Attributtyp und einem oder mehreren Attributwerten. Bei der Definition des Attributtyps wird u.a. festgelegt, ob ein oder mehrere Werte dieses Attributs gespeichert

5 Vgl. http://www.w3.org/2002/ws/ und die dort referenzierten Spezifikationen.
6 Vgl. http://www.w3c.org.
7 Vgl. [RFC 3377], in welchem alle zum LDAPv3-Standard zugehörigen Spezifikationen aufgelistet werden.
8 Hier ist zum einen der zu den LDAP-Spezifikationen gehörende [RFC 2256], sowie die in [RFC 2798] spezifizierte Objektklasse inetOrgPerson zu nennen.
9 Vgl. z.B. [RFC 2307].
10 Zu LDAP-Speichermodellen für X.509-Zertifikate, siehe [Gietz 2003b].
11 Einen Überblick über standardisiertes LDAP-Schema bietet zukünftig die Directory Schema Registry, http://www.schemareg.org., vgl. auch [Gietz 2003a]
12 Für eine kurze Einführung in LDAP, vgl. [Gietz 2002]. Aktuelle Monografien zu LDAP sind [Arkills 2003], [Carter 2003], [Donley 2003] und [Klünther 2003]. Immer noch grundlegend: [Howes 1999] und für das aus X.500 stammende Informationsmodell: [Chadwick 1994]. Eine praktische Zusammenstellung aller relevanten LDAP-RFCs bietet [Loshin 2000].

werden können, welche Syntax diese Werte befolgen müssen, und nach welchen Vergleichsregeln die Werte in Suchvorgängen gefunden werden sollen (z.B. ob Groß-Kleinschreibung berücksichtigt werden soll oder nicht). Außerdem lässt sich eine Attribut-Hierarchie, und somit eine Attribut-Vererbung spezifizieren, wodurch es möglich wird, in Suchvorgängen ein übergeordnetes Attribut (z.B. *name*) anzugeben, um zu erreichen, dass in allen davon abgeleiteten Attributen gesucht wird (z.B.: *commonName, surName, organizationName)*. Ein Attribut *objectclass* spezifiziert welche Objektklassen den Eintrag modellieren, also welche weiteren Attribute in dem Eintrag existieren müssen, bzw. dürfen. Auch Objektklassen stehen in einer Vererbungshierarchie, sodass z.B. die Objektklasse *organizationalPerson* alle Eigenschaften der Objektklasse *person* erbt.
Einträge werden in einer hierarchischen Baumstruktur (*Directory Information Tree,* DIT) eingeordnet, welcher den gesamten von einem Server vorgehaltenen Namensraum abbildet. Mit einem oder mehreren Attributtyp-Wert-Paaren aus denen der sog. *Relative Distinguished Name* (RDN) gebildet wird, erhält ein Eintrag einen Namen, der in der Hierarchieebene, in welcher der Eintrag eingeordnet ist, eindeutig sein muss. Durch die Aneinanderreihung der einzelnen RDNs in den verschiedenen Hierarchie-Ebenen von einem theoretischen Wurzelknoten bis hin zum RDN des Eintrags, wird der sog. *Distinguished Name* (DN) gebildet, welcher ein im gesamten Datenbestand eindeutiger Name ist.
Neben dem Informationsmodell und dem Operationsmodell gehören zum LDAP-Standard u.a.:

- LDIF, *LDAP Data Interchange Format*, [RFC 2849] ein ASCII-Format, mittels dessen man LDAP-Daten bequem austauschen kann.
- LDAP URL [RFC 2255], ein URL-Format, welches die gesamte reiche Funktionalität der Search-Operation abbildet.
- de facto sogar eine Bibliothek für C und Java, da alle Hersteller von SDKs die beiden entsprechenden Internet-Drafts implementiert haben.

LDAP wird traditionell verwendet, um ein Mitarbeiter- oder Benutzerverzeichnis zu pflegen, um z.B. Kontaktdaten der Personen, wie Telefonnummer oder Emailadresse verfügbar zu machen. Im Folgenden soll gezeigt werden, dass sich ein solches Verzeichnis auch für einen zentralen Authentifizierungsdienst nutzen lässt.

3 Authentifizierung in LDAP

Wie bereits erwähnt, ist in dem in [RFC 2251] spezifizierten Netzwerk-Protokoll auch ein Authentifizierungsprozess definiert. Es handelt sich um die sogenannte *bind* –Operation, mittels derer sich ein LDAP-Client beim LDAP-Server anmeldet. Diese Operation ist in ein *bind request* und ein *bind response* aufgeteilt. Der *bind request* enthält neben der LDAP-Protokollversionsnummer (3) einen DN eines Eintrags, welcher die zu beweisende Identität kennzeichnet, also z.B. ein Personeneintrag, sowie schließlich die Kennzeichnung der verwendeten Authentifizierungsmethode und die dazu gehörenden Daten. Hierbei sind augenblicklich zwei Methoden definiert:

- Simple Bind
  hierbei wird zusätzlich zum DN ein Passwort mitgeschickt. Dieses wird vom Server mit dem im durch den DN bezeichneten Eintrag gespeichertem Passwort verglichen. Da das Passwort ungeschützt über das Netz geschickt wird, also jederzeit abgehört werden kann, wird von der Verwendung von *simple bind* ohne vorherige Verschlüsselung der Verbindung abgeraten. Eine solche Verschlüsselung ist jedoch mit TLS möglich und ebenfalls im Standard spezifiziert (s.u.).

Simple bind wird auch für anonymen Zugriff auf LDAP-Verzeichnisse verwendet, wobei sowohl der DN, als auch das Passwort leer bleiben. Alternativ kann bei anonymen Zugriff auch ganz auf die bind-Operation verzichtet werden.

- SASL Bind
  SASL (Simple Authentication and Security Layer) ist ein IETF-Standard ([RFC 2222]), mit dem man den Authentifizierungs-Vorgang von der eigentlichen Anwendung in eine eigene Schicht kapselt, sodass nicht für jedes Anwendungsprotokoll ein eigener Authentifizierungs-Mechanismus definiert werden muss. Neben einer Reihe von verschiedenen Authentifizierungs-Mechanismen wird in SASL auch eine optionale Verschlüsselung der Verbindung zur Verfügung gestellt. Folgende SASL-Mechanismen wurden entweder in [RFC 2222] oder in Folgespezifikationen definiert:
  - PLAIN, ein einfacher Passwortmechanismus, der nicht besser ist als das *simple bind*
  - KERBEROS_V4, wobei Kerberos Version 4[13] verwendet wird, welches jedoch wegen einiger Sicherheitsmängel[14] als obsolet zu betrachten ist. Kerberos ermöglicht Single Sign On (SSO), also einen einmaligen Authentifizierungs-prozess, der für mehrere Authentifizierungsvorgänge gültig ist, da einem Client als Authentifizierungsnachweis ein sog. Kerberos Ticket ausgestellt wird, welches eine bestimmte Gültigkeitsdauer hat und automatisch an Kerberos-fähige Anwendungen übermittelt wird.
  - GSSAPI, ein SASL-Mechanismus, der wiederum eine eigene GSSAPI genannte gekapselte Schnittstelle verwendet, durch die eine Reihe von weiteren Authentifizierungsmechanismen zur Verfügung gestellt werden. GSSAPI steht für *Generic Security Service Application Program Interface* und wird in seiner Version 2 in [RFC 2078] spezifiziert. Die beiden wichtigsten GSSAPI-Mechanismen sind:
    - Kerberos V5 ([RFC 1510]), eine Kerberosversion, in der die erwähnten Sicherheitsschwächen von Kerberos V4 behoben wurden.
    - X.509 [X.509] basierte sogenannte *strong authentication*, bei der die Authentifizierung über X.509-Schlüsselpaare in Verbindung mit der Zertifizierung des öffentlichen Schlüssels geschieht. Über die "Hintertür" GSSAPI wird so das für X.500 spezifizierte *strong bind* ermöglicht.
  - DIGEST MD5, ein in [RFC 2831] spezifizierter SASL-Mechanismus, der auf dem MD5-Hash-Algorithmus [RFC 1321] beruht, bei dem in einem *Challenge-Response*-Verfahren der Server einen Zufallstext an den Client schickt. Dieser errechnet einen auf diesem Text und das Passwort basierenden Hash, den er an den Server zurückschickt. Dieser berechnet ebenfalls den Hash und kann dadurch feststellen, ob der Client im Besitz des richtigen Passwortes ist, ohne dass dieses über das Netz geschickt werden musste.
  - EXTERNAL, ein SASL-Mechanismus, bei dem eine bereits auf unteren Protokollschichten (z.B. via X.509 etablierte gesicherte Verbindungen auf der Transportschicht mittels IPSec [RFC 2401], oder auf der Sessionschicht mittels SSL/TLS [RFC 2246]) etablierte Authentifizierung für die Anwendungsschicht verwendet wird.

Abhängig vom gewählten SASL-Mechanismus werden bei der LDAP-Authentifizierung im *bind request* anstelle des DNs andere Identitätskennungen (z.B. ein Kerberos-principle) und

13 Ursprünglich aus dem MIT-Projekt Athena entstanden, vgl. [Steiner 1988].

14 Vgl. hierzu [Bellovin 1991].

andere Identitätsbeweise mitgeschickt. Ebenfalls abhängig vom Authentifizierungsmechanismus schickt der Server als Antwort unterschiedlich modellierte *bind responses* zurück. Der hierbei mitgegebene Error-Code (SUCCESS wenn die Authentifizierung erfolgreich war, oder aber der Grund, warum diese nicht erfolgreich war) ist jedoch in jedem Fall festgelegt.
In [RFC 2830] wird spezifiziert, wie TLS zur Verschlüsselung der gesamten Kommunikation zwischen Client und Server im LDAP-Protokoll verwendet werden soll. Hierbei initiiert der Client eine solche Verschlüsselung durch den StartTLS-Befehl, wonach die zu verwendende TLS/SSL-Version ausgehandelt wird, sowie die X.509-Zertifikate ausgetauscht werden. Mit TLS kann sich nicht nur der Server mit seinem Zertifikat authentifizieren, sondern auch der Client. In diesem Fall kann die Client-Authentifizierung über den SASL-Mechanismus EXTERNAL auch für die LDAP-Authentifizierung verwendet werden. Ansonsten dient TLS nur zur Verschlüsselung der Verbindung.
[RFC 2829] spezifiziert, welche dieser Authentifikationsmechanismen von einer standardkonformen LDAPv3-Implementierung in jedem Fall unterstützt werden muss. Diese sind:

- anonyme Authentifizierung (keine Authentifizierung, oder *simple bind* mit leerem Passwort)
- Passwortauthentifizierung mittels des SASL-Mechanismus DIGEST MD5
- durch TLS geschützte Passwortauthentifizierung mittels *simple bind* oder TLS geschützte Authentifizierung mittels des SASL-Mechanismus EXTERNAL

Viele LDAP-Implementierungen unterstützen aber mehr als diese drei Pflichtmechanismen. So lässt sich zum Beispiel mit der Open Source Implementierung OpenLDAP[15] eine Authentifizierung mittels Kerberos 5 via SASL-GSSAPI realisieren. Welche SASL-Mechanismen ein Server unterstützt wird in einem speziellen Eintrag, dem sog. *rootDSE*-Eintrag veröffentlicht.
Passwörter[16] können in LDAP-Servern sowohl im Klartext, als auch verschlüsselt abgelegt werden. Das Standard-Schema spezifiziert zur Speicherung des Passworts das Attribut *userPassword.* Wird es verschlüsselt abgelegt, wird der jeweilige Verschlüsselungs- bzw. Hash-Algorithmus in geschwungener Klammer vor das verschlüsselte Passwort gesetzt. Die üblichsten Algorithmen sind crypt, md5 und sha, aber auch die sichereren smd5 und ssha können eingesetzt werden. Damit der Client beim Setzen eines neuen Passworts nicht wissen muss, wie der Server das Passwort ablegt – bei manchen Authentifizierungsverfahren über SASL werden die Passwörter auch gar nicht im LDAP-Server, sondern an anderen Orten gespeichert – wurde in [RFC 3062] eine Erweiterung des LDAP-Protokolls spezifiziert, die bewirkt, dass der Server das vom Client mit dem alten Passwort zusammen geschickte neue Passwort entsprechend seiner Konfiguration verarbeitet und speichert. Des Weiteren wurde in [RFC 3112] ein neues Attribut *authPassword* mit einer eigenen Syntax für verschlüsselte Passwörter spezifiziert.

4 LDAP für Authentifizierungs- und Autorisierungsprozesse in Anwendungen

Nachdem es im Netzwerkprotokoll von LDAP, wie beschrieben, sichere und genau standardisierte Authentifizierungsmechanismen gibt, lag es nahe, dass Anwendungen direkt auf diese zugreifen, anstelle die komplizierten Authentifizierungsschichten wie SASL oder GSSAPI selber zu implementieren. Hierzu muss folgender mittels vorhandener LDAP-Bibliotheken sehr leicht programmierbarer Prozess implementiert werden:

15 http://www.openldap.org.
16 Zu Passwörtern in LDAP vgl. auch [Ströder 2001].

1. Wenn der LDAP-Server keinen anonymen Zugang zulässt, muss sich die Anwendung selbst mit einer Bind-Operation an einem dedizierten LDAP-Eintrag authentifizieren. Diese Authentifizierung kann dann für beliebig viele Anfragen verwendet werden.
2. Die Anwendung erhält von dem zu authentifizierenden Benutzer meist eine LoginId anstelle eines LDAP-DNs. Deshalb muss die Anwendung erst anhand dieser ID den LDAP-Eintrag suchen.
3. Schließlich führt die Anwendung eine Bind-Operation an dem durch die Search-Operation ermittelten Eintrag mit dem vom Benutzer mitgegebenen Passwort durch. Nach dem Erfolg dieser Bind-Operation kann der Benutzer als authentifiziert gelten.
4. Nach Beendigung aller Abfragen kann sich die Anwendung mit einem *unbind* abmelden, falls sie sich in Schritt 1 authentifiziert hat

Viele Anwendungen besitzen bereits eine solche integrierte LDAP-Schnittstelle für die Authentifizierung von Benutzern, wodurch eine anwendungsspezifische und proprietäre Benutzerverwaltung durch eine zentrale Benutzerverwaltung ersetzt werden kann. Alle derartigen Anwendungen können also auf dieselben Benutzerdaten zugreifen und Authentifizierung, sowie Autorisierung über diese abwickeln. Als Beispiel hierfür sei das Modul mod_auth_ldap[17] des Webservers Apache[18] genannt, mittels dessen man sowohl die Authentifizierung der Benutzer, also deren Identitätsfeststellung, als auch die Autorisierung, also die Festlegung, welcher Benutzer auf welche Webseiten zugreifen kann, über LDAP abwickeln kann.
Bei der sog. Authentifizierungsphase, welche genau wie oben beschrieben implementiert wurde, werden folgende Konfigurationsparameter berücksichtigt:

- AuthLDAPURL, eine LDAP-URL, die den LDAP-Servernamen und Port spezifiziert, sowie den BaseDN, und die Suchtiefe ("sub" für den gesamten Teilbaum, "one" für nur eine Hierarchieebene unter dem BaseDN). Außerdem wird an der Stelle der URL, an der normalerweise die zurückzugebenden Attribute angegeben werden (im Folgenden "<attr>" genannt), das LDAP-Attribut bezeichnet, in dem der vom Benutzer angegebene Username (im Folgenden "<username>" genannt) gesucht werden soll. Zusätzlich kann man noch einen LDAP-Filter mit angeben, der mit dem automatisch von mod_auth_ldap gebildeten Filter mit logischem UND kombiniert wird. Der automatisch gebildete Filter besteht aus (<attr=username>). Gibt man in der URL zusätzlich den Filter (objectclass=posixAccount) an, wird mod_auth_ldap folgenden Filter verwenden: (& (objectclass=posixAccount) (<attr>=<username>))
- AuthLDAPBindDN, ein optionaler DN, an dem sich mod_auth_ldap vor der Such-Operation authentifizieren kann.
- AuthLDAPBindPassword, das zu diesem BindDN gehörige Passwort.

Bei der Autorisierungsphase, werden folgende Konfigurationsparameter berücksichtigt:

- require, eine auch ohne mod_auth_ldap gebrauchter Konfigurationsparameter erhält hier Erweiterungen. Folgende Statements lassen sich mit require durchführen:
  - require valid-user: Hierbei erhält jeder Zugriff, der sich erfolgreich am LDAP-Server unter den durch AuthLDAPURL gegebenen Bedingungen authentifiziert hat.
  - require user <Benutzername>: Hier wird jeder einzelne berechtigte Benutzer angegeben, wobei das unter AuthLDAPURL angebene Suchattribut (<attr>) die hier angegebenen Werte enthalten muss.

17 Vgl. die Dokumentation unter http://cvs.apache.org/viewcvs.cgi/*checkout*/httpd-ldap/docs/mod_auth_ldap.html.

18 Das Modul mod_auth_ldap gehört ab der Apache-Version 2.0.41 zur Standard-Distribution.

- require group <Gruppenname>: Hier erhält jeder Zugriff, der in einer mit einem DN bezeichneten Gruppe Mitglied ist, also im Gruppeneintrag in den Attributen member bzw. uniquemember mit seinem DN genannt wird. Folgende weitere Konfigurationsparameter verändern das Verhalten bei require group:
  - AuthLDAPGroupAttributeIsDN on|off: Wenn dieser Schalter auf off gestellt wird, wird anstelle des DN des Gruppenmitglieds, der durch das Attribut <attr> bezeichnete Benutzer in den Werten der Attribute member und uniquemember gesucht.
  - AuthLDAPGroupAttribute: Hiermit kann man andere Attribute als member oder uniquemember angeben, in denen dann anstelle dieser nach Gruppenmitgliedern gesucht wird.
- require dn: Hier werden wiederum einzelne Benutzer angegeben, wobei sie mit ihrem DN, anstelle des Werts des Attributs <attr> bezeichnet werden.

Der Apache-Webserver verfügt noch über ein zweites LDAP-Modul (mod_ldap), welches einen Cache für die LDAP-Suchen zur Verfügung stellt, sowie ein Management der LDAP-Verbindungen, durch welches weniger Bind-Vorgänge notwendig sind und für jede Suche LDAP-Verbindungen bereitgehalten werden. Beide diese Features können die Performance der LDAP-Authentifizierung erheblich steigern.[19]

5 LDAP für Authentifizierung bei Login-Prozessen

Auch bei Login-Prozessen in Betriebssystemen können auf zentrale, durch einen LDAP-Server vorgehaltene Benutzerdaten zugegriffen werden. Bei Unix-Systemen geschieht die Integration der LDAP-Authentifizierung sehr elegant durch die Verwendung von zwei abgekapselten Bibliotheksschichten NSS und PAM, die in Abbildung 1 schematisch dargestellt werden.

Name Service Switch (NSS) ist eine Schicht innerhalb der C-Libraries des Betriebssystems verschiedener Unix-Plattformen, die es ermöglicht, Informationsdienste, die normalerweise Dateien unter /etc als Datenbasis haben (z.B. Accountdaten in /etc/passwd) mit verschiedenen Datenbank-Backbones zu versorgen. Dies geschieht vollkommen transparent. Neben NIS und DNS kann anstelle dieser Dateien auch ein LDAP-Verzeichnis verwendet werden[20], um Daten, wie z.B. Benutzerdaten, Gruppendaten, IP-Diensten, etc. vorzuhalten. Das entsprechende LDAP-Schema ist in [RFC 2307] spezifiziert.

Pluggable Authentication Modules (PAM)[21] ist ein Rahmenwerk für Login-Dienste, welches in vier Module aufgeteilt ist: 1.) Authentifizierung, 2.) Session-Aufbau und –Loggen, 3.) Accountverwaltung mit Login-Policies, sowie 4.) Passwortpolicies. Diese 4 Funktionalitäten können durch PAM mit verschiedenen Technologien verwaltet werden können. Eine der möglichen Technologien für den Authentifizierungsvorgang ist LDAP[22], alternativ kann aber auch z.B. Kerberos, oder sogar SMB (Server Message Block), das Authentifizierungs-protokoll von Windows verwendet werden. Mit PAM wird also eine weitere Abstraktions-schicht eingeführt, die es ermöglicht, allen Anwendungen, die PAM unterstützen, per Konfiguration eine Authentifizierungsmethode zuweisen zu können. Nicht für alle Unix-

19 Vgl. hierzu die Dokumentation unter http://cvs.apache.org/viewcvs.cgi/*checkout*/httpd-ldap/docs/mod_ldap.html?rev=1.1.

20 Vgl. http://www.padl.com/OSS/nss_ldap.html.

21 Vgl. http://www.kernel.org/pub/linux/libs/pam.

22 Vgl. http://www.padl.com/OSS/pam_ldap.html.

Plattformen steht eine PAM-Bibliothek zur Verfügung, aber mindestens für Linux, FreeBSD, OpenBSD, NetBSD, Solaris, HP-UX und AIX.

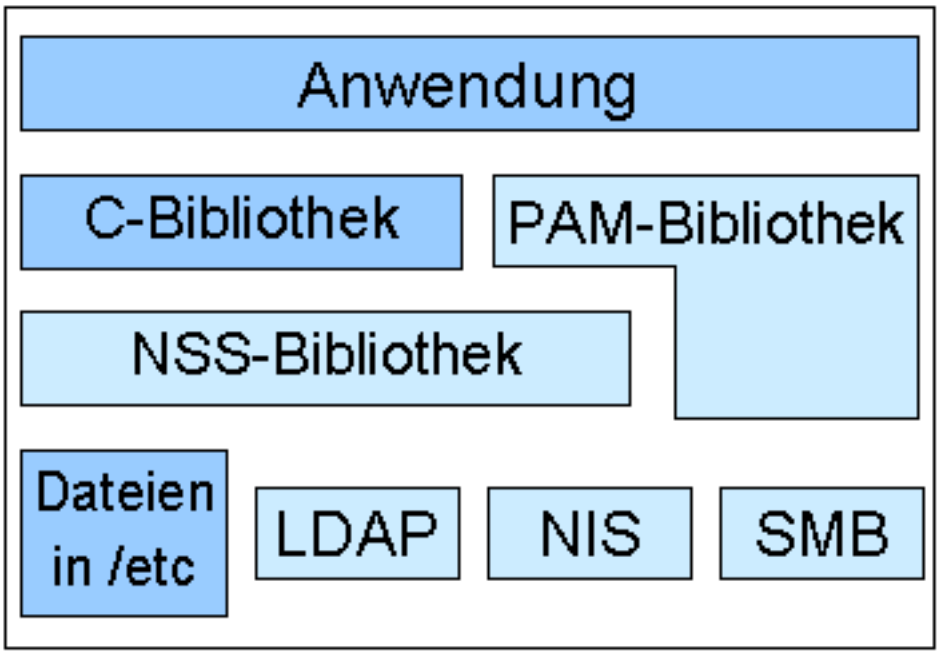

Abbildung 1: Authentifizierungsschichten in Unix

Mit einem zentralen LDAP-Verzeichnis lässt sich über PAM ein zentraler Authentifizierungsdienst aufbauen, wobei ein Benutzer für beliebig viele Rechner ein einziges Passwort benötigt, was nicht nur für den Benutzer von Vorteil ist, sondern auch für den Administratoren, der die Benutzer ebenfalls nur an einer zentralen Stelle managen muss. Das häufige Rücksetzen des Passworts, weil der Benutzer es vergessen hat wird so ebenfalls verhindert.

Auch für den Login-Prozess bei Windows-Rechnern kann eine solche LDAP-basierte Benutzerverwaltung verwendet werden. Dies wird durch den Einsatz von Samba[23] ermöglicht, einer Open-Source-Software, die unter Unix Dienste zur Verfügung stellt, die über das Windows-spezifische Protokoll SMB kommunizieren. Ein solcher Dienst ist neben der Bereitstellung von Dateien und Druckern über das Netz auch der Authentifizierungsdienst, über den sich Windows-Clients an einer Windows-Domäne anmelden. Samba verfügt ebenfalls über eine LDAP-Schnittstelle, sodass wieder die Benutzer-Account-Daten für den Login-Prozess aus der zentralen LDAP-Benutzerverwaltung verwendet werden können.

Als Alternative zu Samba kann die Software pGina[24] auf den Windows-Clients installiert werden, die ebenfalls einen Windows-Login-Prozess u.a. über LDAP-Authentifizierung ermöglicht.

6 Ein Unified Password Szenario

Mit den in den zwei letzten Kapiteln vorgestellten Technologien wurde für den Produktionsbetrieb in einer Firma ein Unified Password System, also ein zentraler

23 Vgl. http://www.samba.org.
24 Vgl. http://pgina.xpasystems.com.

Authentifizierungsdienst aufgebaut, wobei nicht nur der Login-Prozess, sondern auch Authentifizierungsvorgänge in Anwendungen von diesem Dienst bedient wurden. Folgende Anwendungen wurden so integriert: Ein IMAP-Email-Server (Cyrus-imapd), ein SMTP-Dienst mit Authentifizierung vor dem Verschicken einer Mail (Sendmail smtp auth), Secure Shell (sshd) und eine Projekt-Management-Software (Tutos).
Der Authentifizierungsdienst wurde redundant auf zwei Linux-Rechnern mittels LDAP-Replikation implementiert, um die Ausfallssicherheit dieses zentralen Dienstes zu erhöhen.. Abbildung 2 zeigt schematisch diese Architektur.
Samba wurde in der Version 2.2.8a verwendet. Das kürzlich erschienene Samba 3.0 hat gegenüber dieser Version einige wichtige Vorteile. So können mit Samba 3.0 Windows-Benutzer ihr Passwort mit dem Windows-Passwort-Änderungs-Dialog ändern und Samba sorgt dafür, dass die entsprechenden Passwort-Attribute im LDAP-Verzeichnis automatisch aktualisiert werden.
Folgende weitere Software-Pakete wurden verwendet, wobei neuere Versionen dieser Pakete ebenfalls funktionieren sollten:

binutils-2.11.90.0.29-15.i386.rpm
gcc-2.95.3 136.i386.rpm
glibc-devel-2.2.4-40.i386.rpm
make-3.79.1-180.i386.rpm
nss_ldap-167-54.i386.rpm
openldap2-2.0.12-33.i386.rpm
openldap2-client-2.0.12-28.i386.rpm
openldap2-devel-2.0.12-28.i386.rpm
openssl-devel-0.9.6b-62.i386.rpm
pam-devel-0.75-78.i386.rpm pam_

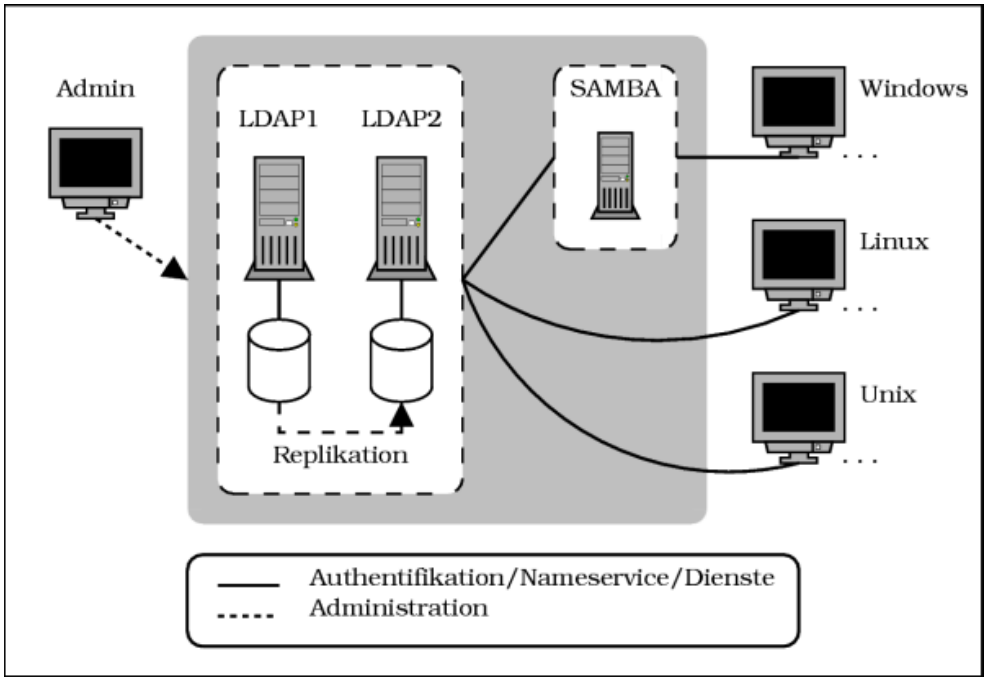

Abbildung 2: Architektur des Unified Login Dienstes

Um die Anzahl der LDAP-Zugriffe zu minimieren, wurde der Name Service Cashing Daemon (nscd) verwendet, wonach keine Performance-Unterschiede gegenüber Systemen, die nicht auf den zentralen LDAP-Dienst zugreifen, festzustellen waren. Hierbei ist zu bedenken, dass auf einem Unix-Client mit NSS_LDAP-Installation, bei jedem Vorgang bei dem die UserId und GroupId ausgelesen werden muss (also z.B. bei jedem ls-Befehl zum Anzeigen von Dateien und deren Besitzer) LDAP-Aufrufe benötigt werden.

7 Sicherheitsrisiken bei der Verwendung von LDAP für zentrale Authentifizierungssysteme

Beim Einsatz eines zentralen Authentifizierungssystem wird die Kompromittierung des zentral gespeicherten Passworts zu einem erhöhten Sicherheitsrisiko. Wenn dieses zentrale Passwort kompromittiert wurde, ist eben nicht nur ein Rechner oder eine Anwendung offen, sondern alle, die am Authentifizierungssystem angeschlossen sind. Deshalb sollten Root-Passwörter immer dezentral auf den jeweiligen Rechnern bleiben und nicht in das LDAP-Authentifizierungssystem integriert werden.

Es gibt mittlerweile eine ganze Reihe von Hackertools, deren dediziertes Ziel es ist, LDAP-Passwörter zu kompromittieren. Eines dieser Tools ist das C-Programm Kold „Knocking on LDAPs Door“ (www.phenoelit.de) welches einen Wörterbuch-Angriff auf jeden Eintrag eines LDAP-Servers durchführt. Ein Perl-Programm mit ähnlicher Funktion heißt LDAP_Brute.pl (angreypacket.com). Solche On-Line-Attacken sind relativ leicht an den Log-Dateien des LDAP-Servers zu erkennen. Darüber hinaus setzen sie voraus, dass man entweder anonym auf den LDAP-Server zugreifen kann, oder dass dem Angreifer mindestens ein Passwort bekannt ist, um überhaupt die Einträge finden zu können. Schließlich funktionieren solche Attacken auch nur bei sehr schwachen Passwörtern, die aus Vornamen oder ganzen Wörtern bestehen. Wenn darauf geachtet wird, dass z.B. mindestens eine Ziffer und ein Sonderzeichen im Passwort enthalten sind, gehen solche Angriffe ins Leere.

Ein weiteres Tool zum Kompromittieren von LDAP-Passwörtern heißt Lumberjack (ebenfalls www.phenolit.de), welches nicht einen LDAP-Server angreift, sondern LDIF-Dateien analysiert. LDIF-Dateien werden öfters zu Replikationszwecken über das Netz geschickt und können, geschieht dies nicht durch verschlüsselte Kanäle, abgefangen werden. Es wurde sogar beobachtet, dass Firmen solche LDIF-Dateien offen im Web abgelegt hatten. Lumberjack versucht die verschlüsselten Passwörter in LDIF-Dateien knacken. Da der Angriff nicht über das Netz geht, der Angreifer also unbegrenzte Zeit für die Analyse hat, versucht Lumberjack jede mögliche Kombination aus Buchstaben, Ziffern und Sonderzeichen, verschlüsselt diese mit dem in geschwungener Klammer angegebenen Algorithmus und vergleicht das Resultat mit den verschlüsselten Passwörtern. Wegen der Existenz solcher Tools, wie Lumberjack ist von einer Klartextübertragung von LDIF-Dateien unbedingt abzuraten. Außerdem sollte darauf geachtet werden, dass die Passwörter in relativ kurzen Abständen erneuert werden.

Ein wesentliches Merkmal für sichere Passwörter ist die Kontrolle der Passwortlänge, der Nutzung eines möglichst großen Zeichenvorrats (Klein und Grossbuchstaben, Ziffer und Sonderzeichen) bei der Bildung von Passwörtern, ihrer Gültigkeitsdauer und der Beschränkung ihrer Wiederverwendung. Eine diese Punkte vorschreibenden Passwort-Regeln (*password policy*) ist gerade im Zusammenhang mit zentralen Authentifizierungssystemen von großer Bedeutung. Viele LDAP-Implementierungen können solche Regeln erzwingen. Ein entsprechender Standard wird augenblicklich im Rahmen der IETF entwickelt [Behera 2003]. Für LDAP-Implementierungen, wie die Open Source Implementierung OpenLDAP, die bisher solche Passwort-Regeln nicht unterstützen, kann diese Funktionalität mittels eines Web-Frontends implementiert werden. Dabei muss aber sichergestellt sein, dass Passwortänderungen nur über dieses Gateway erfolgen. Außerdem muss zusätzlich ein Client regelmäßig überprüfen, ob Passwörter bereits abgelaufen sind und solche Einträge dann entsprechend sperren.

Wie bei allen anderen Netzwerkprotokollen, bilden Angriffsszenarien, wie "Man in the Middle Attack" und das Abhören von Netzverbindungen weitere mögliche Gefahren. Wenn für die LDAP-Kommunikation, insbesondere beim Authentifizierungsvorgang TLS verwendet wird, sollten aber auch solche Angriffe nicht fruchten. Während der Initiierung einer TLS-Verbindung kann sowohl eine Server- als auch eine Client-Authentifizierung

mittels X.509-Zertifikaten durchgeführt werden, wodurch auch "Man in the Middle" Angriffe ausgeschlossen werden können.

8 Zusammenfassung und Ausblick

Auch im DFN-Umfeld, in dem oft große heterogene Computerlandschaften betrieben werden, lassen sich zentrale Authentifizierungssysteme sinnvoll einsetzen. Gerade wenn die im Beitrag beschriebenen Sicherheitsaspekte beachtet werden, lassen sich auf diese Weise leicht administrierbare Benutzerverwaltungen realisieren, welche zu erheblichen Arbeitseinsparungen führen können. Angesichts der zunehmenden Benutzerzahlen und der immer größer werdenden Belastungen der Systemadministratoren werden solche Einsparpotentiale immer wichtiger. Bis auf die dort noch fehlende Implementierung von *Password Policy* kann die Open-Sorce-Software OpenLDAP allen Sicherheitsanforderungen genügen und somit zum Einsatz empfohlen werden.

9 Literaturverzeichnis

[Arkills 2003] Arkills, B., LDAP Directories Explained – An Introduction and Analysis, Boston etc. 2003

[Behera 2003] Behera, P., Poitou, L., Sermersheim, J., Password Policy for LDAP Directories, <draft-behera-ldap-password-policy-06.txt>, March 2003 (work in progress).

[Bellovin 1991] Bellovin, S. M., Merrit, M., Limitations of the Kerberos Authentication System. In: Proceedings of the Winter USENIX Conference, Dallas 1991

[Carter 2003] Carter, G., LDAP System Administration, Sebastopol etc. 2003

[Chadwick 1994] Chadwick, D., Understanding X.500 – The Directory, London 1994

[Donley 2003] Donley, C., LDAP – Programming, Management and Integration, Greenwich 2003

[Gietz 2002] Gietz, P., Verzeichnisdienste für Hochschulen auf Open Source Grundlage. In: Tagungsband Informations- und Verzeichnisdienste in Hochschulen, Hrsg. von J. v. Knop und F. Bode, Düsseldorf 2002, S. 111-127.

[Gietz 2003a] Gietz, P.: Survey of previous work on directory schema registry related technologies and existing LDAP Schema, TERENA Project Directory Schema Registry, Deliverable B, Version 0.9, 31.1.2003. http://www.daasi.de/services/SchemaReg/Schema-reg-Del-B-v0.91.pdf

[Gietz 2003b] Gietz, P., Neues Konzept für einen DFN-weiten Zerti_katsserver. In: Security, E-Learning, E-Services, 17. DFN-Arbeitstagung über Kommunikationsnetze, Hrsg. von J. v. Knop, W. Haverkamp und E. Jessen, Düsseldorf 2003, S. 459-466.

[Howes 1999] Howes, T., Smith, M. C., Good, G. S., Understanding and Deploying LDAP Directory Services, Macmillan Network Architecture and Development Series, o.O. 1999

[Klünther 2003] Klünther, D., Laser, J., LDAP verstehen – OpenLDAP einsetzen, Heidelberg 2003

[Loshin 2000] Loshin, P., Big Book of Lightweight Directory Access Protocol (LDAP)

RFCs, San Diego etc. 2000.

[OMG 2002] Object Management Group, Common Object Request Broker Architecture: Core Specification, Version 3.0.2, December 2002, http://www.omg.org/docs/formal/02-12-02.pdf

[OpenGroup 2002] The Open Group: Business Scenario: Identity Management, 15. July 2002, http://www.opengroup.org/dif/projects/im-scen/idmbs_1.pdf

[Pohlman 2003] Pohlman, M., LDAP Metadirectory Provisioning Methodology – A step by step method to implementing LDAP based metadirectory provisioning & identity management systems, New York etc. 2003

[RFC 1321] Rivest, R., The MD5 Message-Digest Algorithm", RFC 1321, April 1992.

[RFC 1510] Kohl, J., Neuman, C., The Kerberos Network Authentication Service (V5), RFC 1510, September 1993.

[RFC 2078] Linn, J., Generic Security Service Application Program Interface, Version 2, RFC 2078, January 1997.

[RFC 2222] Myers, J., Simple Authentication and Security Layer (SASL), RFC 2222, October 1997.

[RFC 2246] Dierks, T., Allen, C., The TLS Protocol Version 1.0, RFC 2246, January 1999.

[RFC 2251] Wahl, M., Howes, T., Kille, S., Lightweight Directory Access Protocol (v3), RFC 2251, December 1997.

[RFC 2255] Howes, T., Smith, M., The LDAP URL Format, RFC 2255, December 1997.

[RFC 2256] Wahl, M., A Summary of the X.500(96) User Schema for use with LDAPv3, RFC 2256, December 1997.

[RFC 2829] Wahl, M., Alvestrand, H., Hodges, J., Morgan, R., Authentication Methods for LDAP, RFC 2829, May 2000.

[RFC 2830] Hodges, J., Morgan, R., Wahl, M., Lightweight Directory Access Protocol (v3): Extension for Transport Layer Security, RFC 2830, May 2000.

[RFC 2831] Leach, P., Newman, C., Using Digest Authentication as a SASL Mechanism, RFC 2831, May 2000.

[RFC 2307] Howard, L., An Approach for Using LDAP as a Network Information Service, RFC 2307, March 1998.

[RFC 2401] Kent, S., Atkinson, R., Authentication framework. Recommendation X.509, RFC 2401, November 1998.

[RFC 2768] Aiken, B., Strassner, J., Carpenter, B., et al., Network Policy and Services: A Report of a Workshop on Middleware, RFC 2768, February 2000

[RFC 2798] Smith, M., Definition of the inetOrgPerson LDAP Object Class, RFC 2798, April 2000.

[RFC 2849] Good, G.: The LDAP Data Interchange Format (LDIF) - Technical Specification, RFC 2849, June 2000.

[RFC 3062] Zeilenga, K., LDAP Password Modify Extended Operation, RFC 3062, February 2001.

[RFC 3112] Zeilenga, K. LDAP Authentication Password Schema, RFC 3112, May 2001.

[RFC 3377] Hodges, J.; Morgan, R.: Lightweight Directory Access Protocol (v3): Technical Specification, RFC 3377, September 2002.

[Steiner 1988] Steiner, J.G., Neuman, C., Schiller, J., Kerberos: An Authentication Service for Open Network Systems. In: Proceedings of the Winter

USENIX Conference, Dallas 1998.

[Ströder 2001] Ströder, M., Geheimnisträger – Passwörter in LDAP-basierten Verzeichnisdiensten. In: IX 10,2001, S. 139-143

[Wessels 2001] Wessels, Duane, Web Caching, Sebastopol 2001

[X.500] ITU-T. Information technology – Open Systems Interconnection – The Directory, Overview of concepts, models and service. Recommendation X.500, International Telecommunications Union, Geneva, 1993

[X.509] ITU-T. Information technology – Open Systems Interconnection – The Directory, Authentication framework. Recommendation X.509, International Telecommunications Union, Geneva, 1993.